Universidad Autónoma de Madrid
Escuela politécnica superior

Universidad Autónoma de Madrid

Grado en Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

Adaptación de algoritmos de acuerdo con información contextual extraída automáticamente

Autor: Javier Santos Gimeno
Tutor: Marcos Escudero Viñolo
Ponente: Jesús Bescós Cano

JULIO 2021

Adaptación de algoritmos de acuerdo con información contextual extraída automáticamente

Autor: Javier Santos Gimeno
Tutor: Marcos Escudero Viñolo
Ponente: Jesús Bescós Cano

Video Processing and Understanding Lab
Dpto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2021

Este trabajo ha sido parcialmente financiado por el gobierno de España a través del proyecto TEC2017-88169-R MobiNetVideo.

Resumen

Desde la llegada de soluciones basadas en el aprendizaje profundo o *Deep Learning* se ha incrementado el rendimiento exponencialmente de las tareas de visión artificial. Lograr que un detector de objetos funcione de forma correcta en distintos escenarios se ha convertido en una difícil tarea. Por eso es indispensable que un detector de objetos tenga una correcta configuración de los umbrales de confianza o decisión.

Para evaluar la posibilidad de establecer de forma automática la configuración de los umbrales de decisión se propone utilizar un segmentador semántico, que proporciona la información contextual necesaria. El segmentador semántico nos proporciona información semántica sobre la imagen o vídeo, es decir, asocia a cada píxel una etiqueta a la clase a la que pertenece.

Para ello, empezamos realizando un estudio del estado del arte acerca de los conceptos básicos de los detectores de objetos y los segmentadores semánticos. A continuación, se describen los distintos modelos de detector de objetos y segmentador semántico que se ha utilizado para el desarrollo del trabajo.

Posteriormente, se explican los entornos de trabajo utilizados y los conjuntos de datos empleados tanto en el trabajo como en los métodos de detección y segmentación. Seguidamente, se explica el procedimiento del desarrollo del trabajo, que incluye, el detector de objetos, el segmentador semántico y la caracterización y clasificación de las detecciones.

Para finalizar, se evalúan y analizan los resultados obtenidos en los experimentos. Para ello, se realiza una comparativa de los resultados obtenidos en el trabajo. Basándonos en estos resultados se llevarán a cabo unas conclusiones objetivas y se propondrán posibles trabajos futuros.

Palabras clave

Detector de objetos, segmentador semántico, IoU, *Atrous Convolution*, *Deep Learning*.

Abstract

Since the advent of deep learning-based solutions, the performance of computer vision tasks has increased exponentially. Setting up an object detector to operate correctly in different scenarios has become a difficult task. Therefore, it is essential for an object detector to have a correct configuration of confidence or decision thresholds.

To achieve this correct configuration of decision thresholds, we study in this work the feasibility of using a semantic segmentator, which provides the necessary contextual information. The semantic segmentator provides semantic information for each image of a video, associating each pixel a label of the class to which it belongs.

To do so, we start by performing a state-of-the-art study of the basic concepts of object detectors and semantic segmenters. Then, we describe the different models of object detectors and semantic segmentators that have been used for the development of the work.

Subsequently, the working environments used and the data sets used, both in the work and in the detection and segmentation methods, are explained. Next, the development of the work is explained, including the object detector, the semantic segmentator and the characterization and classification of the detections.

Finally, the results obtained in the experiments are evaluated and analyzed. For this purpose, a comparison of the results obtained by different combinations of detector and segmentator is carried out. Based on these results, objective conclusions are drawn and possible future work is proposed.

Key Words

Object detector, semantic segmentator, IoU, Atrous Convolution, Deep Learning.

Agradecimientos

En primer lugar, quiero dar las gracias a mi tutor, Marcos Escudero Viñolo, por ayudarme durante todo el desarrollo del Trabajo de Fin de Grado y por la dedicación y amabilidad que tiene con todos sus alumnos.

Quiero expresar mi agradecimiento a mis padres por haberme brindado la oportunidad de estudiar con la tranquilidad de saber que cuento con su respaldo. También dar las gracias a mis abuelos por su ternura y apoyo durante esta etapa.

Gracias también a Paula por haber confiado en mí y por el apoyo durante estos dos años, ya que ha sido una pilar muy importante en los momentos más difíciles.

Finalmente, gracias a todas las personas que han aparecido en mi vida durante esta etapa, ya que me llevo amigos para toda la vida. Y gracias a todos los compañeros por su cooperación y ayuda durante la carrera, ya que sin ellos no habría sido posible.

Javier Santos Gimeno

Julio 2021

Índice general

Índice de Figuras

Índice de Tablas

1 Introducción

1.1. Motivación

Las tareas de visión artificial han incrementado su rendimiento exponencialmente desde la llegada de soluciones basadas en el aprendizaje profundo o *Deep Learning*. La configuración de los detectores de objetos se ha vuelto una tarea compleja para que funcione en diversos escenarios.

Sin embargo, todavía es un requisito indispensable para su correcto funcionamiento tener una buena configuración de los umbrales de decisión o confianza. Estos umbrales suelen establecerse de forma global, es decir, un mismo umbral para toda la imagen o el vídeo analizados. Es por ello, que establecer un umbral de confianza global hará que haya zonas de la imagen en las que el rendimiento sea correcto mientras que en otras zonas caiga mucho el rendimiento.

Por eso se propone establecer diferentes umbrales en función de las zonas de la escena donde aparezcan detecciones. Estas zonas son inviables anotarlas de forma manual, ya que estas variarían si la cámara vibrara o se moviese.

La información contextual proporcionada por un segmentador semántico podría ser útil para identificar estas zonas de forma automática, asociando a cada píxel una etiqueta correspondiente a la clase a la que pertenece. Esta técnica no está exenta de errores, pero es interesante ver como se relacionan la distribución de clases semánticas en una detección de personas con la calidad de detección.

1.2. Objetivos

El principal objetivo de este Trabajo de Fin de Grado es estudiar la viabilidad de adaptación de algoritmos a través de información contextual obtenida de manera automática. Esta información contextual se obtendrá mediante la segmentación semántica de la escena.

El objetivo de este trabajo es evaluar la viabilidad de utilizar información contextual adquirida automáticamente a partir de la segmentación semántica para predecir la bondad de las detecciones de personas obtenidas mediante un detector de objetos del estado del arte. Si este estudio es exitoso, en el futuro se podría plantear fijar umbrales distintos en función de

la zona en la que aparezca la detección para poder así eliminar falsas detecciones y mejorar la efectividad de reconocimiento de imágenes.

Como objetivo adicional planteamos la posibilidad de seleccionar la mejor combinación de métodos de detector y segmentador atendiendo a los estadísticos resultantes del objetivo anterior. Del mismo modo, también usaremos estos estadísticos para medir el incremento del rendimiento para los diferentes umbrales de decisión o confianza.

Estos objetivos pueden dividirse en varios objetivos parciales:

- Análisis y estudio del estado del arte.
- Evaluación de varios detectores de objetos y segmentadores semánticos.
- Caracterización de las detecciones y entrenamiento de la red neuronal para clasificar detecciones en base a la distribución de información semántica en su interior.
- Evaluación experimental de las combinaciones exploradas y de los clasificadores entrenados.

1.3. Organización de la memoria

El presente trabajo se organiza de la siguiente manera:

- Capítulo 1. Introducción del Trabajo de Fin de Grado en el que se definen las motivaciones y los objetivos.
- Capítulo 2. Se explican los conceptos básicos de los métodos de detección de objetos y segmentación semántica. También se analizan en mayor detalle los distintos modelos empleados en el trabajo de cada uno de los métodos
- Capítulo 3. Se visualizan cada uno de los entornos de desarrollo con los que el trabajo ha sido realizado. Por otro lado, se hará un análisis de las bases de datos utilizadas, tanto para el trabajo como para los métodos de detección y segmentación. Por último, se explican las librerías de métodos empleadas como base de ambos métodos.
- Capítulo 4. Se define el procedimiento que se ha seguido a lo largo del trabajo y se explica en detalle cada una de las etapas que lo componen.
- Capítulo 5. Se realizan y evalúan los experimentos partiendo de las bases de datos y procedimientos mencionados anteriormente donde se representan y se discuten gráficamente los resultados.
- Capítulo 6. Conclusiones acerca de los resultados obtenidos y posibles trabajos futuros a implementar.
- Bibliografía.

2 Estado del arte

2.1. Introducción

En esta parte del trabajo se van a describir los métodos de detección de objetos y de segmentación semántica evaluado en este TFG. Para ello, se hará un introducción de cada método, explicando las principales características, y a continuación se describirán los modelos empleados para cada uno de los métodos.

Los modelos empleados para la segmentación semántica son PSPNet, DeepLabV3, y su versión mejorada, DeepLabV3+. Para la detección de objetos se emplean los modelos de Yolact y R-CNN.

2.2. Segmentación semántica

2.2.1. Introducción a la segmentación semántica

El éxito de las técnicas de aprendizaje profundo en diversas tareas de visión por ordenador de alto nivel, en particular, los enfoques supervisados como las redes neuronales convolucionales (CNNs) para la clasificación de imágenes o la detección de objetos, motivó a los investigadores a explorar las capacidades de dichas redes para los problemas de etiquetado a nivel de píxel como la segmentación semántica.

La clave de estas técnicas de aprendizaje profundo, que les da una ventaja sobre los métodos tradicionales, es la capacidad de aprender representaciones de características apropiadas para el problema en cuestión.

Actualmente, las técnicas de aprendizaje profundo más exitosas para la segmentación semántica provienen de un precursor común: *Fully Convolutional Network* (FCN) [1]. La idea de este enfoque era aprovechar las CNN existentes como potentes modelos visuales capaces de aprender jerarquías de características. Transformaron esos modelos de clasificación existentes y conocidos, como ResNet [2], en modelos totalmente convolucionales, sustituyendo las capas totalmente conectadas por otras convolucionales para obtener mapas espaciales en lugar de puntuaciones de clasificación. Estos mapas se muestrean mediante convoluciones fraccionadas (también llamadas deconvoluciones [3]) para producir salidas densas etiquetadas por píxel.

Figura 2.1: Ejemplo de segmentación semántica.

Este trabajo se considera un hito, ya que mostró cómo las CNNs pueden ser entrenadas de principio a fin para este problema, aprendiendo eficientemente a hacer predicciones densas para la segmentación semántica con entradas de tamaños arbitrarios.

Este enfoque logró una mejora significativa en la precisión de la segmentación con respecto a los métodos tradicionales en conjuntos de datos estándar como PASCAL VOC [4], preservando al mismo tiempo la eficiencia en la inferencia. Por todas estas razones, y otras contribuciones significativas, el modelo FCN es la piedra angular del aprendizaje profundo aplicado a la segmentación semántica

2.2.2. Integración del conocimiento del contexto

La segmentación semántica es un problema que requiere la integración de información procedente de varias escalas espaciales. También implica que haya un equilibrio entre la información local y la global. Por un lado, la información local o de grano fino es crucial para lograr una buena precisión a nivel de píxel. Por otro lado, también es importante integrar la información del contexto global de la imagen para poder resolver las ambigüedades locales.

Las CNNs que suelen usarse para clasificación de imágenes tienen dificultades para lograr este equilibrio. Las capas de agrupamiento, que permiten a las redes alcanzar cierto grado de invariabilidad espacial y mantener a raya el coste computacional, eliminan la información del contexto global. Incluso las CNN puras, es decir, sin capas de agrupación, están limitadas.

Se pueden adoptar diferentes enfoques para que las CNNs puedan hacer uso de esa información global. Algunos de estos enfoques pueden ser:

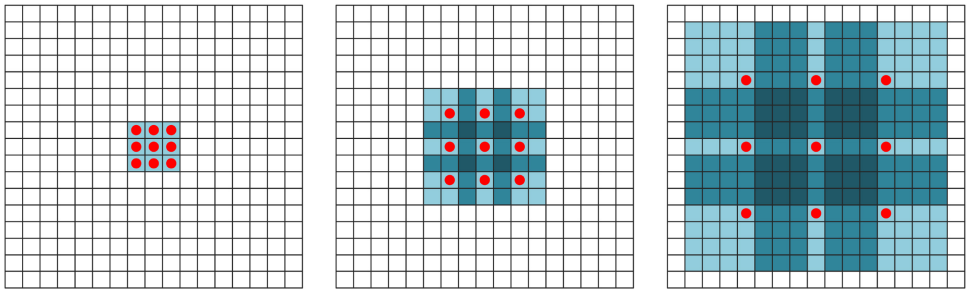
- **Campos aleatorios condicionales**: Como se ha mencionado antes, la invariabilidad espacial de las arquitecturas CNN [2] limita la misma precisión espacial para las tareas de segmentación. Un enfoque posible para refinar el resultado de un sistema de segmentación y potenciar su capacidad para captar detalles de grano fino es aplicar una etapa de pos procesamiento utilizando un campo aleatorio condicional (CRF).

  Los CRF permiten combinar la información de bajo nivel de la imagen con los resultados de los sistemas multiclase que producen puntuaciones de clase por píxel. Esta combinación es especialmente importante para captar la información global y los detalles locales más finos.

- **Convoluciones dilatadas**: Las convoluciones dilatadas, también denominadas Atrous Convolutions, son una generalización de los filtros convolucionales con factor de Kronecker [5] que permiten la expansión de los campos receptivos sin perder resolución. En otras

palabras, las convoluciones dilatadas son las que hacen uso de filtros sobremuestreados. La tasa de dilatación l controla ese factor de sobremuestreo.

Como se muestra en la Figura 2.2, el apilamiento de la convolución dilatada hace que los campos receptivos crezcan exponencialmente mientras que el número de parámetros de los filtros mantiene un crecimiento lineal. Esto significa que las convoluciones dilatadas permiten una eficiente extracción de características densas en cualquier resolución arbitraria. También es importante señalar que las convoluciones típicas son sólo convoluciones dilatadas en 1.



(a) 1-dilated (b) 2-dilated (c) 3-dilated

Figura 2.2: Filtros de convolución dilatados con varias tasas de dilatación. Extraído de [6]

- **Predicción multiescala**: Otra posible forma de abordar la integración del conocimiento del contexto es el uso de predicciones multiescala. Casi todos los parámetros de una CNN [2] afectan a la escala de los mapas de características generados. En otras palabras, la misma arquitectura tendrá un impacto en el número de píxeles de la imagen de entrada que corresponden a un píxel del mapa de características. Esto significa que los filtros aprenderán implícitamente a detectar características a escalas específicas (presumiblemente con cierto grado de invariancia).

  Además, esos parámetros suelen estar estrechamente vinculados al problema en cuestión, lo que dificulta la generalización de los modelos a diferentes escalas. Una posible forma de superar ese obstáculo es utilizar redes multiescala que, por lo general, hacen uso de múltiples redes que se dirigen a diferentes escalas y luego fusionan las predicciones para producir una única salida.

- **Fusión de características**: Otra forma de añadir información de contexto a una arquitectura totalmente convolucional para la segmentación es la fusión de características. Esta técnica consiste en fusionar una característica global (extraída de una capa anterior de una red) con un mapa de características más local extraído de una capa posterior. Las arquitecturas habituales, como la FCN [1] original, utilizan conexiones entre capas no consecutivas para realizar una fusión tardía combinando los mapas de características extraídos de diferentes capas.

2.3. Modelos de Segmentación semántica

2.3.1. PSPNet

PSPNet, o *Pyramid Scene Parsing Network* [7], es un modelo de segmentación semántica que utiliza *Pyramid Pooling Module* (PPM) que explota la información de contexto global mediante la agregación de contexto basada en diferentes regiones. Las pistas locales y globales hacen que la predicción final sea más fiable.

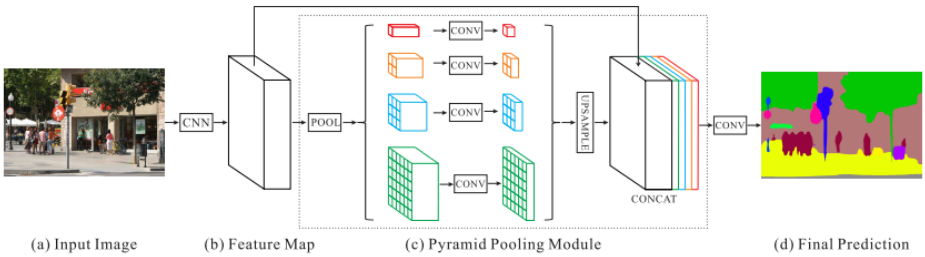


Figura 2.3: Resumen del modelo PSPNet. Extraído de [7].

Con el PPM, se propone la red de análisis de escenas piramidales (PSPNet), como se ilustra en la Figura 2.3 . Dada una imagen de entrada, PSPNet utiliza una CNN pre entrenada [2] con la estrategia de red dilatada para extraer el mapa de características. El tamaño final del mapa de características es de 1/8 de la imagen de entrada (b). Encima del mapa, se utiliza el PPM para recopilar información de contexto. Utilizando una pirámide de 4 niveles, los núcleos de agrupación cubren la totalidad, la mitad y pequeñas porciones de la imagen. Se fusionan para crear la prioridad global. A continuación, concatenamos la prioridad global con el mapa de características original en la parte final de (c). Por último, una capa de convolución genera el mapa de predicción final (d).

Para explicar dicha estructura, PSPNet proporciona una contextualización global muy eficaz para el análisis de la escena a nivel de píxel. PPM puede recoger niveles de información, más representativos que la agrupación global [8].

2.3.2. DeepLabV3

Los modelos basados en redes totalmente convolucionales (FCN) [1] han demostrado una mejora significativa en varios parámetros de segmentación. Hay varias variantes de modelos propuestos para explotar la información contextual para la segmentación, incluyendo los que emplean entradas multiescala (es decir, pirámide de imágenes) o los que adoptan modelos gráficos probabilísticos (como DenseCRF con un algoritmo de inferencia eficiente [9]). En este caso, hablamos principalmente del modelo que utiliza la estructura codificador-decodificador.

Normalmente, las redes de codificador-decodificador contienen un módulo codificador que reduce gradualmente los mapas de características y captura la información semántica superior, y un módulo decodificador que recupera gradualmente la información espacial. Partiendo de esta idea, se puede utilizar DeepLabv3 [10] como módulo codificador y añadir un módulo decodificador sencillo pero eficaz para obtener segmentaciones más nítidas.

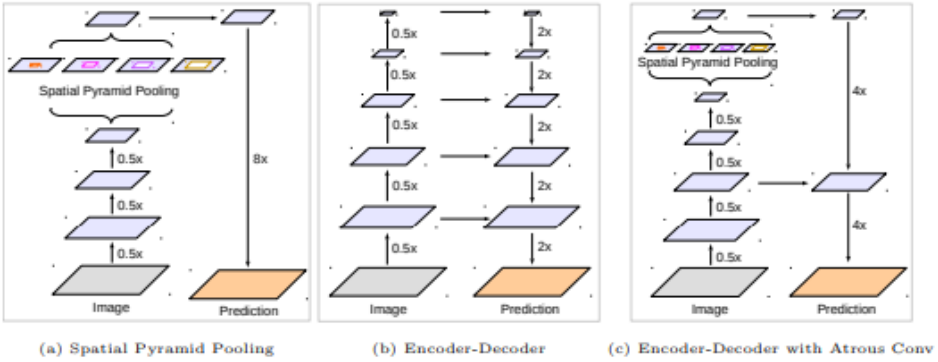
DeepLabV3 utiliza un codificador-decodificador con *Atrous Convolution*. La *Atrous Convolution* es una poderosa herramienta que nos permite controlar explícitamente la resolución de las características calculadas por *Deep Convolutional Neural Networks* (redes neuronales convolucionales profundas) y ajustar el campo de visión del filtro con el fin de capturar la información multiescala que generaliza la operación de convolución estándar.

DeepLabv3 [10] emplea la *Atrous Convolution* para extraer las características calculadas por DCNN (*Deep Convolutional Neural Networks*) a una resolución arbitraria.

Además, DeepLabv3 aumenta el módulo de *Atrous Spatial Pyramid Pooling* (ASPP), que sondea las características convolucionales a múltiples escalas aplicando la *Atrous Convolution* con diferentes tasas, con las características a nivel de imagen.

2.3.3. DeepLabV3+

El módulo de SPP (*Spatial pyramid pooling*) o la estructura de codificación-decodificación se utilizan en las redes neuronales profundas para la tarea de segmentación semántica. Los módulos SPP son capaces de codificar información contextual multiescala sondeando las características entrantes con filtros u operaciones de agrupación a múltiples velocidades y múltiples campos de visión efectivos. Por otro lado, la estructura de codificación-decodificación permite que las segundas redes pueden capturar límites de objetos más nítidos recuperando gradualmente la información espacial.



(a) Spatial Pyramid Pooling (b) Encoder-Decoder (c) Encoder-Decoder with Atrous Conv

Figura 2.4: DeepLabV3+ mejora a DeeplabV3, que emplea SPP (a), con un modulo decodificador (b). Extraído de [11]

En concreto, DeepLabv3+, combina las ventajas de ambos métodos y amplía DeepLabv3 añadiendo un módulo decodificador simple pero eficaz para refinar los resultados de la segmentación, especialmente a lo largo de los límites del objeto, como vemos en la Figura 2.4.

También se puede aplicar la *Atrous Convolution* para extraer las características del codificador a una resolución arbitraria, dependiendo de los recursos de computación disponibles. Además, se explora el modelo Xception[12] y se aplica la convolución separable en profundidad a los módulos *Atrous Spatial Pyramid Pooling* y decodificador, dando como resultado una red codificadora-decodificadora más rápida y potente.

2.4. Detección de objetos

2.4.1. Introducción al detector de objetos

La detección de objetos (ilustrada en la Figura 2.5) es un problema fundamental y desafiante de la visión por ordenador que ha sido un área de investigación activa durante varias décadas. El objetivo de la detección de objetos es determinar si hay alguna instancia de objetos de determinadas categorías (como humanos, coches, bicicletas, perros o gatos) en una imagen y, si está presente, devolver la ubicación espacial y la extensión de cada instancia de objeto.

Como piedra angular de la comprensión de imágenes y de la visión por ordenador, la detección de objetos constituye la base para resolver tareas de visión complejas o de alto nivel, como la segmentación, la comprensión de escenas o el seguimiento de objetos.

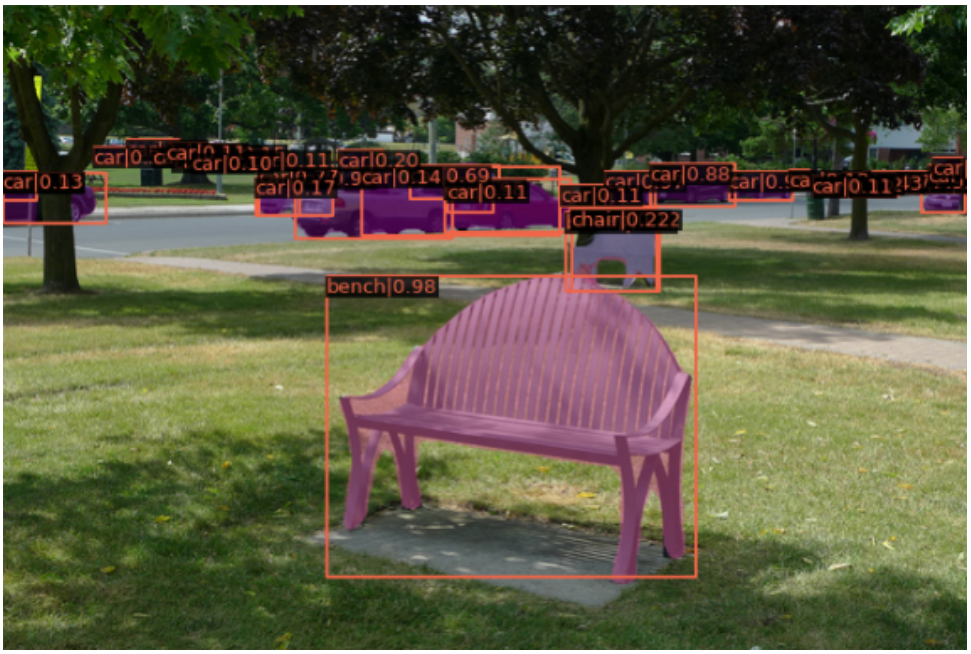


Figura 2.5: Ejemplo de un detector de objetos.

2.4.2. Tipos de detector de objetos

Los detectores de objetos de imagen preexistentes se pueden dividir en dos categorías, una es el detector de dos etapas, el más representativo, Faster R-CNN [13]. La otra es el detector de una etapa, como YOLO [14]. Los detectores de dos etapas tienen una alta precisión de localización y reconocimiento de objetos, mientras que los detectores de una etapa logran una alta velocidad de inferencia.

Las dos etapas de los detectores de dos fases pueden ser divididas por la capa de agrupación *RoI* (Región de Interés). Por ejemplo, en Faster R-CNN, la primera etapa, denominada RPN, una red de propuesta de regiones, propone cuadros delimitadores de objetos candidatos. En la segunda etapa, las características se extraen mediante la operación *RoIPool* (RoI Pooling) para las tareas de clasificación y regresión de cajas delimitadoras [15]. La Figura 2.6 (a) muestra la arquitectura básica de los detectores de dos etapas.

Además, los detectores de una etapa proponen los recuadros predichos a partir de las imágenes de entrada directamente sin el paso de propuesta de región, por lo que son eficientes en tiempo y pueden utilizarse para dispositivos en tiempo real. La Figura 2.6 (b) muestra la arquitectura básica de los detectores de una etapa.

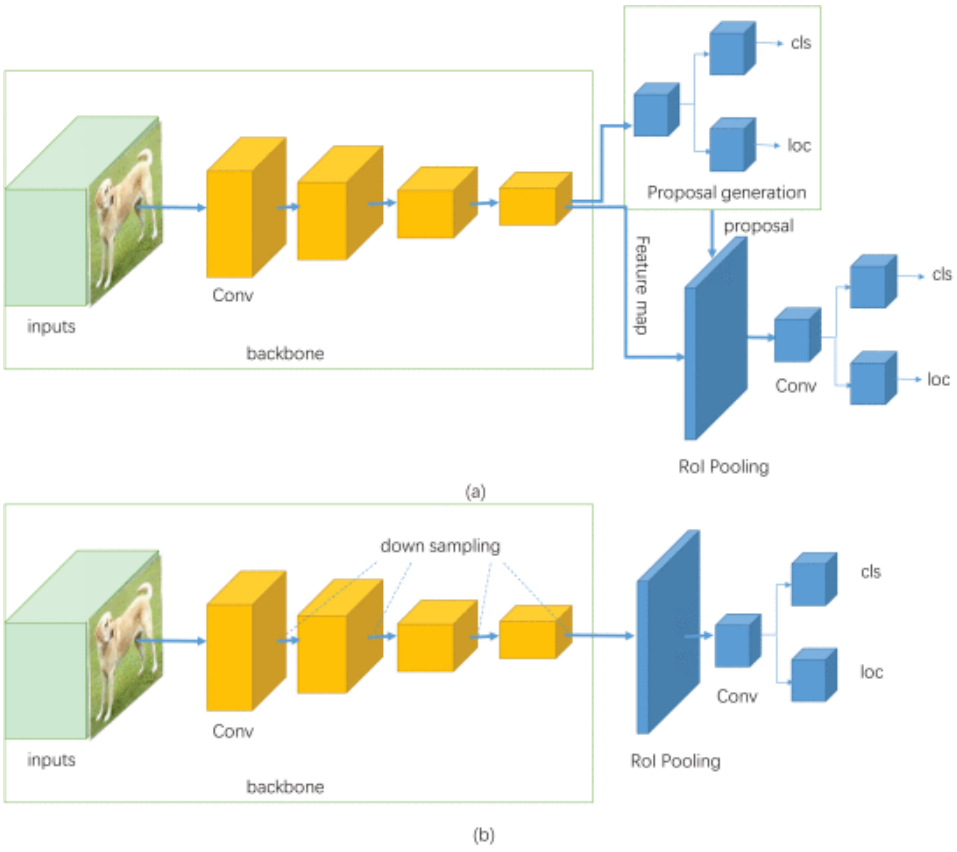


Figura 2.6: Estructura de los dos tipos de detector de objetos. Extraído de [16]

2.5. Modelos de detección de objetos

2.5.1. R-CNN

El sistema de detección R-CNN consta de cuatro módulos. El primer módulo genera propuestas de regiones independientes de la categoría. El segundo módulo extrae un vector de características de longitud fija de cada propuesta de región. El tercer módulo es un conjunto de SVM lineales específicas de cada clase para clasificar los objetos de una imagen. El último módulo es un regresor de cajas delimitadoras para la predicción precisa de las mismas.

Primero, para generar propuestas de regiones, se adopta el método de búsqueda selectiva. A continuación, se utiliza una CNN para extraer un vector de características de dimensión igual a 4096 de cada propuesta de región. Como la capa totalmente conectada necesita vectores de entrada de longitud fija, las características de las propuestas de regiones deben tener el mismo tamaño. Se adopta un tamaño fijo de 227x227 píxeles como entrada de la CNN.

Como sabemos, los objetos de las distintas imágenes tienen diferentes tamaños y relaciones de aspecto, lo que hace que las propuestas de región extraídas por el primer módulo tengan un tamaño diferente. Independientemente del tamaño o de la relación de aspecto de la región candidata, todos los píxeles se redimensionan en un cuadro delimitador apretado a su alrededor con el tamaño requerido de 227x227.

La red de extracción de características consta de cinco capas convolucionales y dos capas totalmente conectadas. Y todos los parámetros de la CNN se comparten en todas las categorías, y cada categoría entrena una SVM independiente.

2.5.2. Yolact

El objetivo de Yolact es añadir una rama de máscaras a un modelo de detección de objetos de una sola etapa, de la misma manera que Mask R-CNN [15] lo hace a Faster R-CNN [13], pero sin un paso explícito de localización de características.

Para ello, se divide la compleja tarea de segmentación de instancias en dos tareas paralelas más sencillas que pueden ensamblarse para formar las máscaras finales. La primera rama utiliza un FCN [1] para producir un conjunto de máscaras prototipo del tamaño de la imagen que no dependen de ninguna instancia. La segunda añade una cabeza adicional a la rama de detección de objetos para predecir un vector de coeficientes de máscara que codifica la representación de una instancia en el espacio de prototipos. Por último, para cada instancia que sobrevive al NMS (*Non-Maximum Suppression*), se construye una máscara para esa instancia combinando linealmente el trabajo de estas dos ramas.

Para el detector de columna vertebral se prioriza la velocidad así como la riqueza de características, ya que predecir estos prototipos y coeficientes es una tarea difícil que requiere buenas características para hacerlo bien. Por lo tanto, el diseño del detector de columna vertebral sigue de cerca al de RetinaNet [17], haciendo hincapié en la velocidad.

Yolact es capaz de lograr resultados de detección similares a los de YOLOv3 [18] a velocidades similares, sin emplear ninguna de las mejoras adicionales de YOLOv2 y YOLOv3 como el entrenamiento multiescala, las cajas de anclaje optimizadas, la codificación de regresión basada en celdas y la puntuación de objetividad. Esto es debido a que las mejoras en el rendimiento de la detección en Yolact provienen principalmente del uso de FPN (*Feature Pyramid Networks*)y del entrenamiento con máscaras.

3
Materiales y Métodos

3.1. Introducción

Tras haber analizado el estado del arte se dará paso a describir brevemente los entornos de desarrollo que se han empleado para el desarrollo del trabajo. A su vez, se verán las bases de datos utilizadas en los experimentos, así como las bases de datos empleadas en cada método. También se verán las bases que se han empleado para realizar el detector de objetos y el segmentador semántico.

3.2. Entornos de desarrollo

3.2.1. PyTorch

PyTorch [19] es una librería de aprendizaje automático para Python utilizada principalmente para el procesamiento del lenguaje natural. El software fue desarrollado por los equipos de inteligencia artificial de Facebook Inc. en 2016. PyTorch ofrece dos características significativas que incluyen el cálculo tensorial y las redes neuronales.

PyTorch utiliza un módulo Autograd para calcular la diferenciación automática. Y gracias a esto, ahorra tiempo en el desarrollo de las redes neuronales, ya que la diferenciación de los datos se realiza rápidamente en el paso previo.

3.2.2. Google Colaboratory

Google Colaboratory (Google Colab) es un entorno de cuadernos Jupyter que es de uso gratuito y no requiere ninguna configuración. Esto permite desarrollar aplicaciones de aprendizaje profundo utilizando librerías como Keras, TensorFlow, PyTorch y OpenCV.

La característica más importante que distingue a Colab de otros servicios gratuitos en la nube es que proporciona GPU donde se aprovecha la potencia de hardware de Google. Esto

permite realizar los trabajos a una gran velocidad y sin importar la potencia del equipo que se disponga.

El entorno de Google Colab ha sido utilizado para realizar las primeras pruebas del trabajo donde se obtuvieron las bases de datos de los detectores de objetos y los segmentadores semánticos.

3.2.3. Matlab

Matlab es una plataforma con un lenguaje de un gran rendimiento para el cálculo técnico. Integra el cálculo, la visualización y la programación en un entorno fácil de usar en el que los problemas y las soluciones se expresan en una notación matemática familiar.

Destaca por ser un sistema interactivo cuyo elemento de datos básico es una matriz que no requiere ser dimensionada. Esto permite resolver muchos problemas de cálculo técnico, especialmente aquellos con formulaciones matriciales y vectoriales, en una fracción de tiempo muy pequeña en comparación con un programa en un lenguaje escalar no interactivo como puede ser C.

Es importante recalcar el uso del paquete de Deep Learning (*Deep Learning ToolBox*), el cual ha sido creado para poder diseñar y crear redes neuronales. En este trabajo se ha usado para entrenar un clasificador que ayudase a decidir la mejor combinación de detector y segmentador.

3.3. Bases de datos

El dataset utilizado para este trabajo es MOT17Det. Este conjunto de datos consta de 14 vídeos, 7 vídeos para el conjunto de prueba (test) y 7 vídeos para el conjunto de entrenamiento (train). Para este experimento se ha centrado la atención en los vídeos de entrenamiento. Estos conjuntos de datos están formados por 6 vídeos de 1920 x 1080 píxeles y un vídeo de 640 x 480 píxeles. Para cada uno de los vídeos, el dataset nos proporciona un fichero de *Ground Truth*, del cual obtenemos todas las detecciones anotadas de cada uno de los frames.

Es importante resaltar que este dataset consta de distintos tipos de clases pero en este trabajo nos centramos únicamente en la detección de la clase de persona.

En la tabla 3.1 se pueden observar los distintos vídeos empleados en el trabajo así como sus características:

- **Muestra:** Figura de un frame de ejemplo de cada vídeo.
- **FPS:** Son los frames por segundo que tiene cada uno de los vídeos.
- **Resolución:** Es el tamaño de cada frame (ancho x alto).
- **Longitud:** Son el número total de frames que tiene cada uno de los vídeos. Entre paréntesis, la duración en tiempo.
- **Detecciones:** Es el número total de detecciones anotadas por cada vídeo.
- **Densidad:** Es el número medio de detecciones que hay por cada frame.

Muestra	Nombre	FPS	Resolución	Longitud	Detecciones	Densidad
	MOT17-13	25	1920x1080	750(00:30)	11642	15.5
	MOT17-11	30	1920x1080	900(00:30)	9436	10.5
	MOT17-10	30	1920x1080	654(00:22)	12839	19.6
	MOT17-09	30	1920x1080	525(00:18)	5325	10.1
	MOT17-05	14	640x480	837(01:00)	6917	8.3
	MOT17-04	30	1920x1080	1050(00:35)	47557	45.3
	MOT17-02	30	1920x1080	600(00:20)	18581	31.0

Tabla 3.1: Características generales del conjunto de datos empleado para los experimentos MOT Challenge 17.

3.3.1. Dataset para segmentación semántica

El dataset con el que han sido entrenados los distintos segmentadores semánticos y que, por lo tanto, define la salida de estos métodos, es el de Cityscapes [20]. Cityscapes es uno de los conjuntos de datos más famosos en el análisis de escenas urbanas. Este conjunto de datos contiene 2975 imágenes para el entrenamiento, 500 imágenes para la validación y 1525 imágenes de prueba. Hay un total de 19 clases disponibles para la tarea de segmentación semántica, aunque en realidad hay 30 clases (como se puede ver en la figura 3.1), pero se ignoran aquellas que son muy raras.

Este conjunto de datos a gran escala contiene un diverso conjunto de secuencias de vídeo estereoscópicas que han sido grabadas en escenas callejeras de 50 ciudades diferentes, con anotaciones de una gran calidad a nivel de píxel de 5000 fotogramas, además de un conjunto mayor de 20000 fotogramas con anotaciones débiles.

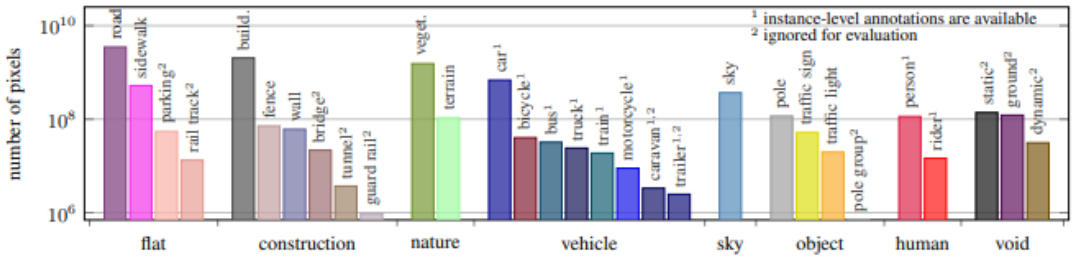


Figura 3.1: Imagen de las clases del dataset Cityscapes. Extraído de [20].

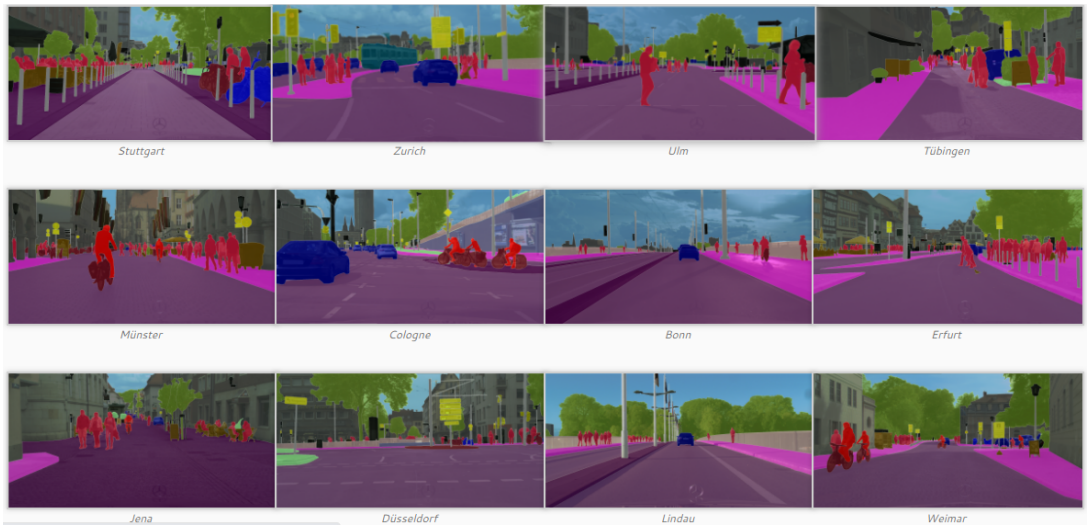


Figura 3.2: Imágenes de algunos ejemplos de distintas ciudades del dataset Cityscapes. Extraído del dataset Cityscapes.

3.3.2. Dataset para detección de objetos

El dataset que se ha utilizado para entrenar diferentes detectores de objetos y que, por lo tanto, define la salida de estos métodos, es el de COCO [21], que contiene 118000 imágenes de entrenamiento, 5000 de validación y 20000 de prueba.

COCO es una de las bases de datos de reconocimiento de objetos de código abierto más populares utilizadas para entrenar programas de *deep learning*. Esta base de datos incluye cientos de miles de imágenes con millones de objetos ya etiquetados para el entrenamiento.

Podría decirse que el elemento más importante del aprendizaje automático supervisado es el acceso a un conjunto de datos grande y bien documentado del que aprender. Patrocinado por Microsoft, COCO segmenta las imágenes en categorías y objetos, a la vez que proporciona subtítulos y etiquetas de contexto legibles por la máquina. Todo ello reduce drásticamente el tiempo de entrenamiento básico para cualquier IA que necesite procesar imágenes.

Este dataset se ha utilizado para realizar todas las detecciones de los detectores de objetos para cada uno de los vídeos.

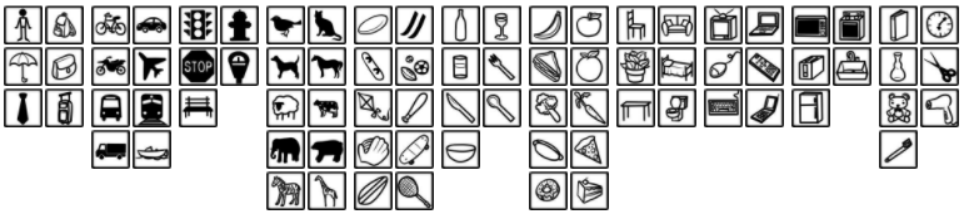


Figura 3.3: Imágenes de las distintas clases del dataset COCO.

3.4. OpenMMLab

El objetivo de este trabajo es juntar dos métodos de reconocimiento de imágenes como son el detector de objetos y el segmentador semántico. Para obtener las salidas de ambos tipos de métodos se ha usado MMDetection [22] y MMSegmentation [23] en el entorno de Google Colab.

MMDetection es una librería de métodos de detección de objetos de código abierto basada en PyTorch. Forma parte del proyecto OpenMMLab. Como características principales tiene, un diseño modular, soporte de múltiples marcos de trabajo y una alta eficiencia.

MMSegmentation es una librería de métodos de segmentación semántica de código abierto basada en PyTorch. Forma parte del proyecto OpenMMLab. Como características principales tiene las anteriormente nombradas en MMDetection y además tiene un parámetro de referencia unificado.

4

Diseño y Desarrollo

4.1. Introducción

Una vez visto los entornos de ejecución y las distintas bases de datos se procederá en este capítulo a realizar el diseño y desarrollo de las pruebas. En primer lugar, vamos a definir la metodología que hemos seguido para realizar las pruebas.

Como se ha comentado en el capítulo anterior, el objetivo del trabajo es juntar los métodos de detección de objetos y de segmentación semántica. El desarrollo del trabajo se ha dividido en distintos pasos como se puede observar en la figura 4.1. Partiendo de una imagen inicial, obtenemos las detecciones que se caracterizarán mediante la segmentación semántica y a continuación se realizará un clasificador para finalmente mediar la capacidad de clasificar las detecciones en correctas e incorrectas en base a las descripciones semánticas.

4.2. Procedimiento

4.2.1. Detector de objetos

El detector de objetos, como se ha explicado anteriormente en 2.4, se encarga de determinar si hay alguna instancia de objetos de determinadas categorías en una imagen, y de devolver su ubicación espacial.

Para el método de detección de objetos se han usado los modelos anteriormente nombrados en la sección 2.5, R-CNN y Yolact, ambos entrenados en el dataset de Coco. Para cada vídeo, almacenamos todas las detecciones de cada frame, con independencia de la confianza del detector. Para ello registramos la posición en ambos ejes, la anchura y altura de la bounding box, el porcentaje de acierto de cada detección y la etiqueta de la clase a la que corresponde. Para la realización de este trabajo nos hemos focalizado en la etiqueta de la clase de persona.

Está parte del procedimiento se realiza en el entorno de Google Colab y es uno de los pasos previos a la caracterización de detecciones. Más adelante habrá que combinar cada modelo de detección con los distintos modelos de segmentación semántica.

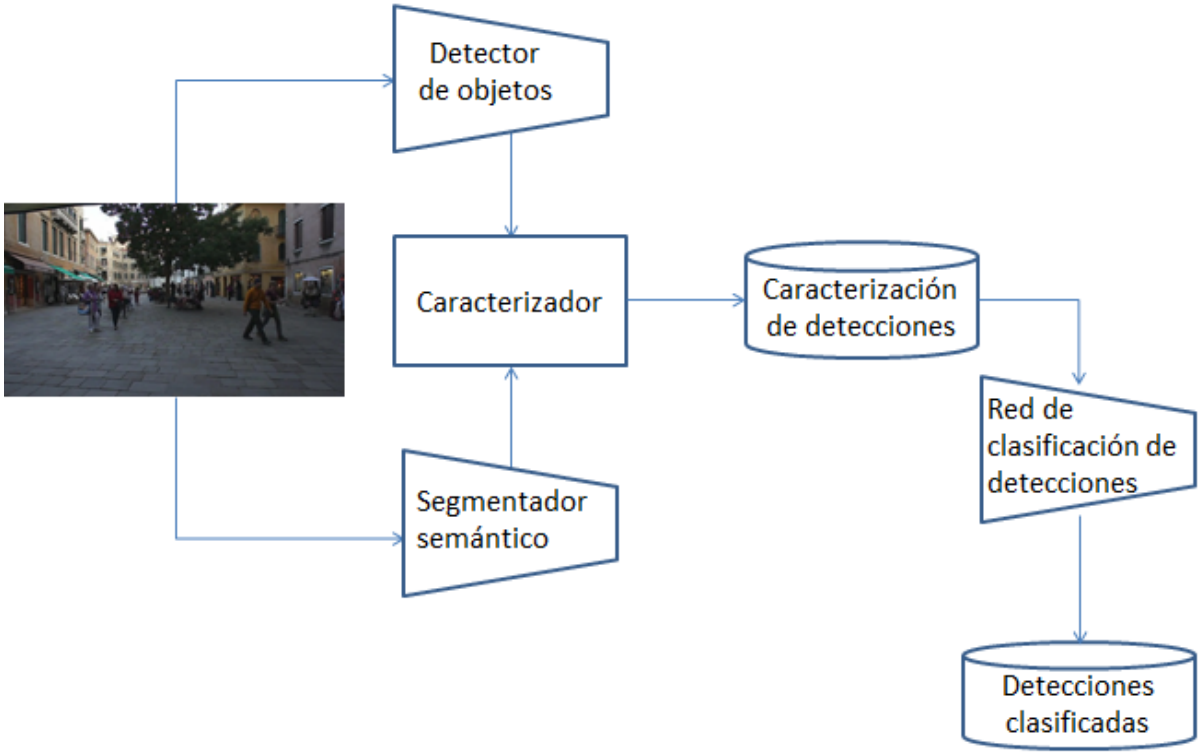


Figura 4.1: Imagen del diagrama de bloques del trabajo.

4.2.2. Segmentador semántico

La principal función de un segmentador semántico, como ya se ha explicado en el capítulo del estado del arte 2, es la de asignar una etiqueta a cada píxel de la imagen de entrada. De esta forma se obtendrá la información semántica que es imprescindible para los siguientes pasos.

Para el método de segmentación semántica se han usado los modelos anteriormente nombrados en la sección 2.3, PSPNet, DeepLabV3 y DeepLabV3Plus, todos ellos entrenados en el dataset de Cityscapes. Para almacenar la información semántica se guarda el mapa semántico de cada frame de los vídeos, es decir, una matriz del tamaño del frame con un índice en cada píxel con la clase semántica asignada por el segmentador utilizado, con independencia de la confianza asignada por el segmentador.

Esta información semántica será de gran utilidad en el proceso del trabajo ya que son los datos que se van a emplear para caracterizar cada una de las detecciones obtenidas con los detectores de objetos.

4.2.3. Caracterizador

La caracterización es un paso clave en el trabajo ya que es el método que fusiona la segmentación semántica con la detección de objetos. Una vez obtenidos todos los datos de detección y segmentación dejamos atrás el entorno de Google Colab para meternos de lleno en el programa Matlab.

Lo primero es cargar todos los datos en el entorno de Matlab. Los ficheros en los que se basa el desarrollo del caracterizador son las detecciones, la información semántica y el ground thruth (GT) de las detecciones. Las detecciones y la información semántica se han obtenido en los dos bloques recientemente explicados, mientras que el GT nos lo proporciona el propio dataset de

los vídeos.

Ahora el objetivo, con estos tres ficheros, es crear un nuevo fichero de texto (base de datos con la caracterización de las detecciones) en el que se incluyan la posición, tamaño y frame de las detecciones, el *Intersection over Union* (IoU) entre detecciones y GT, y un histograma normalizado con la información semántica que hay dentro de la caja de cada detección. De esta forma, se busca caracterizar las detecciones, es decir, para cada detección tenemos toda su información semántica correspondiente.

Para calcular el IoU, se compara la caja de cada detección con todas las cajas del GT, se asocia cada detección con el GT con el que obtenga valor de IoU más alto. También se ha realizado de la forma inversa, es decir, comparando cada caja del GT con todas las cajas de las detecciones, aunque esa información no será utilizada en este TFG, podrá usarse como parte del trabajo futuro. El IoU es la intersección entre el GT y la detección. En la figura 4.2 se puede observar como se calcula.

Como tenemos dos detectores y tres segmentadores, tenemos que realizar la caracterización anteriormente explicada, con el IoU y el histograma semántico, para cada vídeo con cada una de las combinaciones, es decir, cada detector con cada segmentador, de forma que obtendríamos un total de 42 ficheros distintos (6 ficheros por cada vídeo).

El objetivo de esto es averiguar cuál es la combinación óptima de segmentador y detector si se busca maximizar el número de detecciones correctas utilizando para ello un clasificador sobre las caracterizaciones semánticas, lo cual se estudiará en el siguiente bloque.

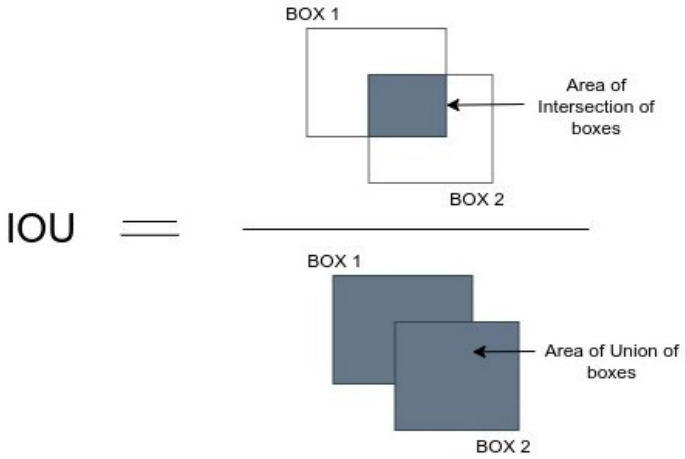


Figura 4.2: Imagen de como se obtiene el IoU.

4.2.4. Red de clasificación de detecciones

Lo siguiente a realizar es un clasificador de detecciones con una red neuronal que nos ayude a decidir la mejor combinación de métodos. En este caso, se ha empleado el paquete de Matlab de Deep Learning (*Deep Learning ToolBox*) que incluye la herramienta nprtool, que es la aplicación que realiza la red neuronal para clasificar los datos. En la figura 4.3 se puede ver como es la interfaz del clasificador.

Para esta red de clasificación se ha elegido el algoritmo de aprendizaje *Scaled Conjugate Gradient Backpropagation* (SCG). SCG es un algoritmo de aprendizaje supervisado para redes neuronales, y es parte de la clase de métodos de gradiente conjugado. SCG puede entrenar cualquier red siempre que sus funciones de peso, entrada de la red y transferencia, tengan

funciones derivadas. El *Backpropagation* se utiliza para calcular las derivadas del rendimiento con respecto a las variables de pesos y así ir alterando los valores de estos pesos para optimizar el valor de la función objetivo según avanza el entrenamiento.

Lo siguiente a realizar, es obtener las variables de input y target que se van a emplear para el clasificador. La variable input, es la variable en la que introducimos los datos con los que vamos a clasificar, es decir, el histograma normalizado de cada detección, que en esta ocasión será una variable con 19 clases para cada detección, ya que estas son las distintas clases que tiene el dataset Cityscapes para la segmentación semántica. En la variable target se asigna la correspondiente decisión a partir del IoU para cada detección, es decir, para cada detección de la variable input, target le asigna un 1 o un 0 en función del valor del IoU.

En conclusión, la red neuronal será un clasificador con el algoritmo de aprendizaje *Scaled Conjugate Gradient Backpropagation* (SCG) en el que se entrenarán una serie de datos con 19 elementos por muestra. El clasificador está formado por 10 capas ocultas y nos dará como resultado final una clasificación de 1 o 0 para cada muestra.

En particular, exploraremos varios valores de IoU, tal y como se describe en el capítulo 5, lo que nos permitirá medir la bondad de la combinación bajo diferentes hipótesis de solapamiento entre detección y GT.

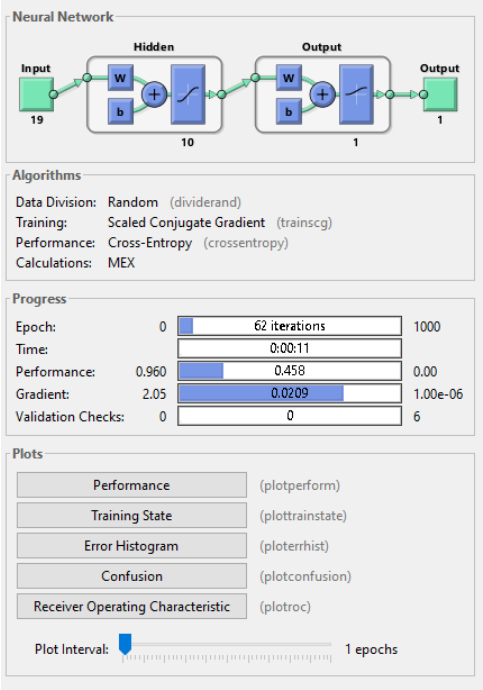


Figura 4.3: Imagen de la interfaz de Nprtool.

5 Experimentos

5.1. Introducción

Es este capítulo se van a explicar los experimentos realizados. Para ello, se comenzará a explicar como se han establecido los umbrales y cual ha sido la división del dataset. A continuación, se evaluarán las medidas de rendimiento y se discutirán los resultados obtenidos en las gráficas.

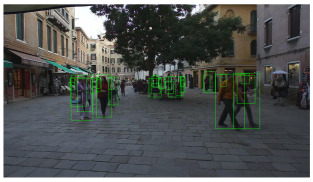
5.2. Entorno experimental

Como se ha comentado en el capítulo anterior, se entrena la red de clasificación según el umbral de IoU que se establece previamente. Para analizar las medidas de rendimiento únicamente se han usado los umbrales comprendidos entre 0.30 y 0.70 ya que el resto de umbrales no aportaban información al estudio, es decir, 9 umbrales distintos que van aumentando en pasos de 0.05.

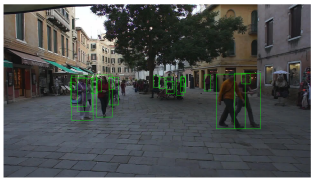
Como se puede observar en la figura 5.1, se ha hecho una comparativa sobre como afecta el IoU a las detecciones en una imagen. En ella se observa que según va aumentando el valor del IoU, el número de detecciones disminuye.

Para las primeras pruebas realizadas, se usó toda la información de todos los vídeos y se dividieron estos datos en entrenamiento (*train*) (70 %), validación (*validation*) (15 %) y prueba (*test*) (15 %). El inconveniente de esto es que el clasificador dividía los datos de forma aleatoria, lo que suponía que no se podría realizar la prueba (*test*) para un vídeo únicamente. Debido a esto hubo que buscar otra manera de clasificar.

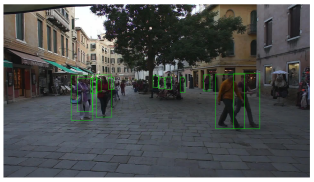
Alternativamente, en vez tener la información de todos los vídeos como argumento de entrada de clasificador, únicamente se emplearon cinco vídeos. Con estos cinco vídeos se entrena el clasificador asignando el 100 % de los datos de estos vídeos a entrenamiento (*train*) y dejando los otros dos vídeos para realizar las pruebas (*test*). En conclusión, se emplean cinco de los



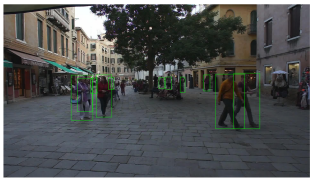
(a) IoU = 0.30



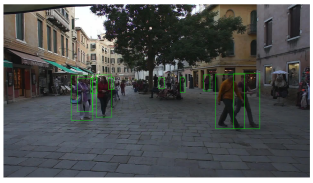
(b) IoU = 0.40



(c) IoU = 0.50



(d) IoU = 0.60



(e) IoU = 0.70

Figura 5.1: Imágenes con sus detecciones, cada una con un umbral sobre el IoU distinto.

siete vídeos para crear la red entrenada y a continuación, se utilizan los dos restantes para las pruebas (test).

Una vez realizado esto, quedan 9 clasificadores distintos, dependiendo del umbral establecido, para cada una de las combinaciones de los dos modelos de detección y los tres de segmentación, para un total de 54 clasificadores.

Las detecciones de los dos vídeos de test se pasan por cada uno de los clasificadores entrenados y se obtienen los valores de la matrices de confusión TP, TN, FP y FN como se puede ver en la tabla 5.1.

	1	**0**
1	TP	FP
0	FN	TN

Tabla 5.1: Matriz de confusión: TP (True Positive), FP (False Positive), FN (False Negative) y TP (True Positive).

5.3. Medidas de rendimiento

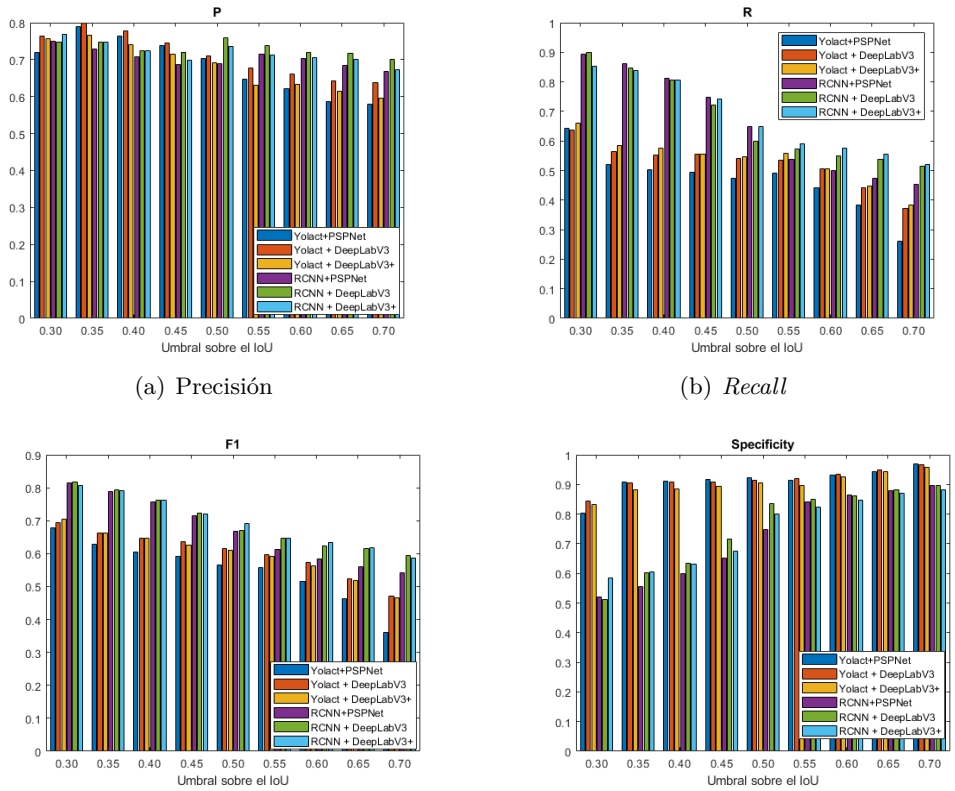
Y como se ha descrito en los capítulos anteriores, vamos a obtener las medidas de rendimiento para cada una de las combinaciones de detectores y segmentadores. De forma que se podrá observar y comparar en las gráficas las variaciones que existen entre las distintas combinaciones.

5.3.1. Medidas de rendimiento de entrenamiento

En la figura 5.2 se pueden observar las gráficas de las medidas de rendimiento correspondientes al entrenamiento. De esta forma se puede ver el rendimiento del clasificador con los datos de entrenamiento (*train*).

Se observa que las medidas de rendimiento para los datos de entrenamiento (*train*) tienen unos buenos valores. Esto tiene sentido ya que en el entrenamiento se introduce, junto a la información semántica, si la detección es o no correcta, por lo que con esta información el clasificador tiene un mejor rendimiento.

En cuanto al *Recall* y al *F1-Score*, se obtienen unos mejores resultados en los métodos que usan como detector de objetos RCNN. Mientras que la especificidad es al contrario, los mejores resultados son de los métodos con Yolact como detector de objetos. En cuanto a la precisión, los resultados son similares para todas las combinaciones de detectores y segmentadores.



(a) Precisión (b) *Recall*

(c) *F1-Score* (d) Especificidad

Figura 5.2: Medidas de rendimiento de la clasificación de los datos de entrenamiento (*train*).

Ahora se da paso a discutir las medidas de rendimiento para el conjunto de datos de prueba (*test*).

5.3.2. Precisión

La precisión es la relación entre la cantidad de detecciones que han sido clasificadas correctamente como positivas y todas las clasificadas como positivas. Se calcula de la siguiente manera:

$$P = \frac{TP}{TP + FP} \tag{5.1}$$

En la imagen 5.3 se pueden observar los datos correspondientes a la precisión para las distintas combinaciones de detector y segmentador y los distintos umbrales sobre el IoU.

En la gráfica se puede observar que todas las combinaciones siguen la misma forma, se mantienen constantes hasta que llega el máximo y entonces comienzan a descender según aumentamos el umbral. En el caso de Yolact con los distintos segmentadores el pico máximo surge entre los umbrales 0.30 y 0.35. Mientras que para el caso de RCNN combinado con los distintos segmentadores, dicho pico máximo se encuentra entre los umbrales 0.50 y 0.55.

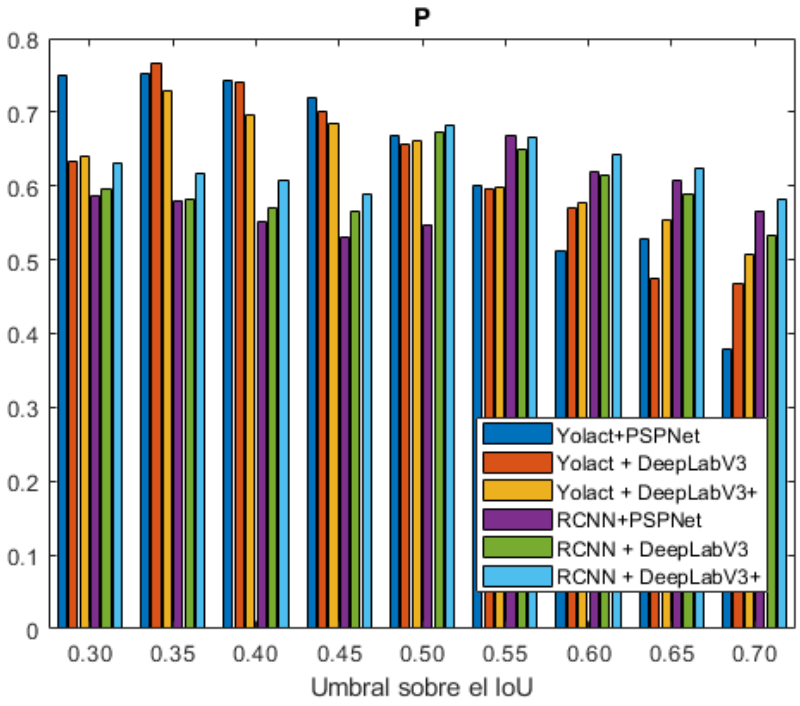


Figura 5.3: Imagen de la representación de la precisión.

5.3.3. Recall

El *Recall* (también conocido como Sensibilidad) de un clasificador es la relación entre la cantidad de datos clasificados correctamente como positivos y la cantidad de datos anotados como positivos en el GT. Se calcula de la siguiente manera:

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

En la imagen 5.4 se pueden observar los datos correspondientes al *Recall* para las distintas combinaciones de detector y segmentador y los distintos umbrales sobre el IoU.

En este caso, todas las combinaciones de métodos siguen el mismo formato descendente según aumenta el valor del umbral, lo cual tiene sentido, ya que cuanto mayor es el umbral más probable es que aumenten los falsos negativos al haber menos detecciones supervivientes. Como se observa en la gráfica, los valores más altos del *Recall* se encuentran en el umbral 0.3 y las mejores combinaciones son las de RCNN + PSPNet y RCNN + DeepLabV3+.

5.3.4. F1-Score

La media armónica de la precisión y el *recall* da una puntuación llamada *F1-Score*, que es una medida del rendimiento de la capacidad de clasificación del modelo. Se calcula de la siguiente manera:

$$F1 = \frac{2TP * TP}{2TP + FP + FN} \tag{5.3}$$

En la imagen 5.5 se pueden observar los datos correspondientes al *F1-Score* para las distintas combinaciones de detector y segmentador y los distintos umbrales sobre el IoU.

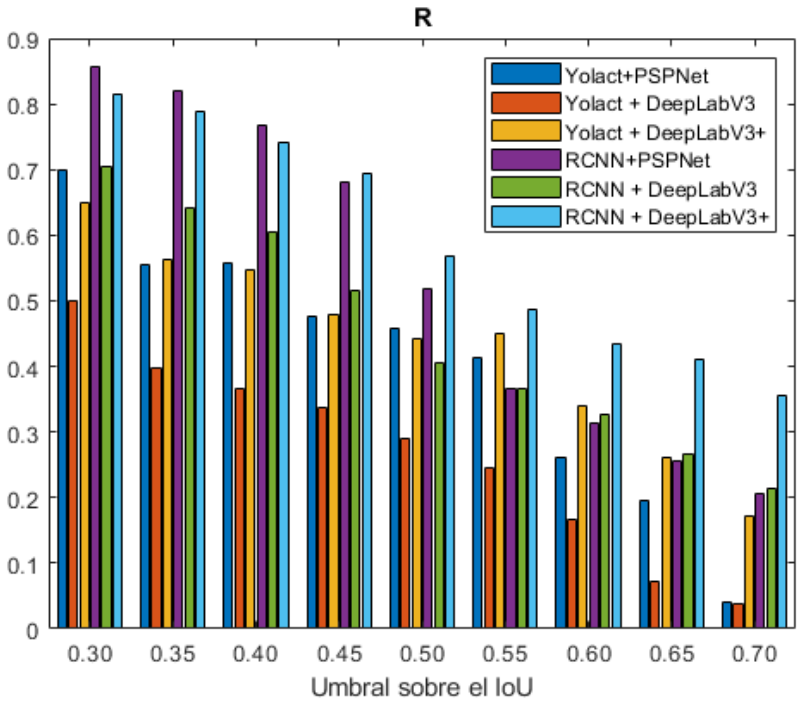


Figura 5.4: Imagen de la representación del *Recall*.

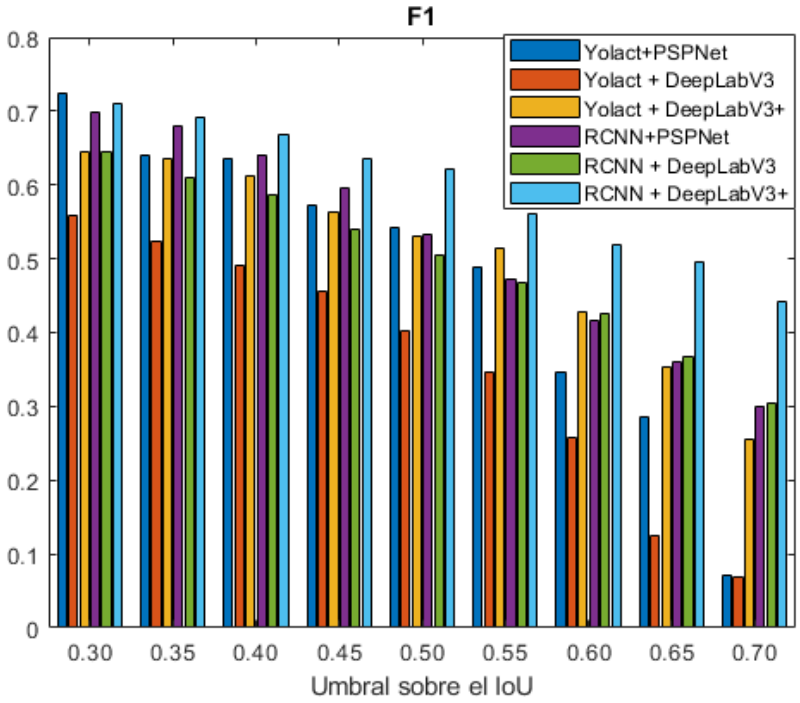


Figura 5.5: Imagen de la representación del *F1-Score*.

Al igual que pasaba en la gráfica del *Recall*, el *F1-Score* sigue una linea descendente según aumenta el umbral. Esto da lugar a que los clasificadores funcionen mejor con umbrales de menor valor. Es importante resaltar el método de RCNN + DeeplabV3+ ya que, aunque sigue una linea descendente, la pendiente con la que desciende es mucho menor que en el resto de combinaciones de métodos.

5.3.5. Especificidad

La especificidad de un clasificador es la relación entre la cantidad de datos clasificados correctamente como negativos y la cantidad de datos anotados como negativos. Se calcula de la siguiente manera:

$$Especificidad = \frac{TN}{TN + FP} \tag{5.4}$$

En la imagen 5.6 se pueden observar los datos correspondientes al especificidad para las distintas combinaciones de detector y segmentador y los distintos umbrales sobre el IoU.

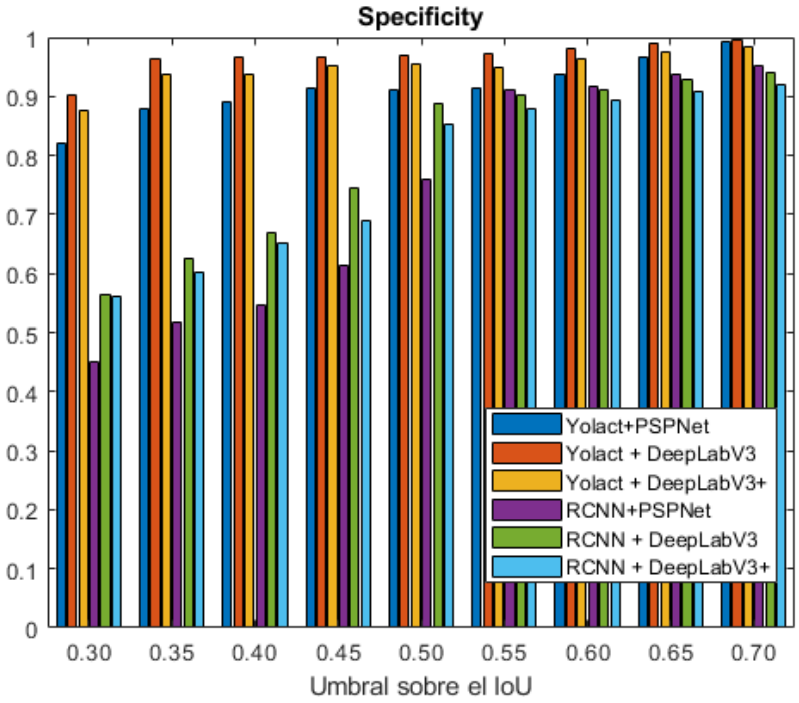


Figura 5.6: Imagen de la representación del especificidad.

En la gráfica de la especificidad se puede observar que sigue una trayectoria ascendente según aumenta el umbral, al contrario de lo que pasaba en el *Recall*. Como se ha comentado antes, según va aumentando el umbral va aumentando el número de ceros, o de negativos. Es por eso, que la especificidad, al medir los datos clasificados correctamente como negativos, vaya aumentando según el umbral aumenta. La combinación de Yolact con los distintos segmentadores, aunque tiene una pendiente ascendente, mantiene todos sus valores en un rango estrecho para todos los umbrales.

5.4. Discusión de los métodos

En esta sección se va a discutir cual es la combinación de los modelos de detector y segmentador óptima. Para comenzar, se ha realizado un resumen de las medidas de rendimiento para cada combinación de métodos, como se puede observar en la tabla 5.2. En ella se recogen los valores más altos para cada combinación de métodos y el umbral al que se han obtenido para los datos de prueba (*test*).

Método	**Especificidad**	**Precisión**	**Recall**	**F1-Score**
Yolact + PSPNet	0.9913 (*0.70*)	0.753 (*0.35*)	0.6984 (*0.30*)	0.7234 (*0.30*)
Yolact + DeepLabV3	0.9959 (*0.70*)	0.7654 (*0.35*)	0.5005 (*0.30*)	0.5591 (*0.30*)
Yolact + DeepLabV3+	0.9839 (*0.70*)	0.7293 (*0.35*)	0.6494 (*0.30*)	0.6448 (*0.30*)
RCNN + PSPNet	0.9508 (*0.70*)	0.6692 (*0.55*)	0.8579 (*0.30*)	0.6975 (*0.30*)
RCNN + DeepLabV3	0.9414 (*0.70*)	0.6719 (*0.50*)	0.7033 (*0.30*)	0.6452 (*0.30*)
RCNN + DeepLabV3+	0.9203 (*0.70*)	0.6833 (*0.50*)	0.816 (*0.30*)	0.7113 (*0.30*)

Tabla 5.2: Tabla con las mejores medidas de rendimiento de cada método. Se indica el valor más alto, de cada combinación de detector y segmentador, para cada una de las cuatro medidas de rendimiento explicadas en este capítulo, y, entre paréntesis, el umbral al que se ha obtenido.

En cuanto a la precisión, las combinaciones de métodos que tienen como modelo de detección Yolact, tienen unos valores más altos en comparación a los métodos con el modelo de detección RCNN. De forma contraria, los valores más altos en el *Recall* están en las combinaciones de métodos que tienen RCNN como modelo de detección.

La especificidad tiene valores más altos en las combinaciones que tienen a Yolact como detector de objetos. El *F1-Score* al depender de la precisión y del *Recall* tiene los valores más altos en los métodos donde estas dos medidas son mayores.

En referencia a los umbrales, todas las combinaciones de métodos tienen sus valores más altos en los mismos umbrales a excepción de la precisión, que para las combinaciones de métodos con Yolact como detector, tienen el umbral en 0.35, mientras que con RCNN como detector, el umbral esta entre 0.50 y 0.55.

Para finalizar, se concluye que las mejores combinaciones de métodos son la de Yolact, como detector de objetos, con PSPNet, como segmentador semántico, y la de RCNN, como detector de objetos, con DeepLabV3+, como segmentador semántico. Esta reflexión se fundamenta en que ambos métodos tienen los valores más altos de *F1-Score*. Entre ambos métodos, se destaca la primera combinación, Yolact + PSPNet, ya que tiene unos valores más altos en la especificidad.

En conclusión, después del estudio y la observación gráfica de los resultados, se propone como mejor combinación de modelos, el método Yolact + PSPNet, Yolact como detector de objetos y PSPNet como segmentador semántico.

6

Conclusiones y Trabajo Futuro

6.1. Conclusiones

En este Trabajo de Fin de Grado se ha tratado de predecir la bondad de las detecciones de un detector de objetos mediante la información contextual obtenida a partir de un segmentador semántico.

El estudio y análisis del estado del arte ha resultado un punto muy útil, ya que gracias a ello se han obtenido conocimientos acerca de los detectores de objetos y los segmentadores semánticos y algunos de los modelos utilizados para el desarrollo del trabajo.

En primer lugar, se han obtenido las detecciones y la información semántica de cada uno de los modelos propuestos. Una vez conseguido, se ha empleado la información semántica para caracterizar cada una de las detecciones. Con las detecciones caracterizadas se ha entrenado una red neuronal para clasificar las detecciones en correctas o incorrectas bajo diferentes hipótesis de solapamiento con el ground-truth.

Una vez clasificadas todas las detecciones se han comparado los resultados de las distintas combinaciones de detectores de objetos con segmentadores semánticos. con lo que se han obtenido 6 combinaciones distintas. Para cada combinación de métodos se han calculado las medidas de rendimiento para distintos umbrales, calculados a partir del IoU.

Una vez obtenidas estas medidas de rendimiento, se ha pasado a discutir cual es la mejor combinación de modelos de detector y segmentador. Finalmente, gracias a los resultados obtenidos, se considera que la mejor combinación de métodos es la de Yolact + PSPNet.

Esto nos lleva al siguiente apartado donde el trabajo realizado da lugar a una serie de posibles trabajos futuros a implementar.

6.2. Trabajo Futuro

Como se ha podido observar, gracias a los resultados, se ha logrado decidir cual es la mejor combinación de detectores y segmentadores. Esto da lugar a propuestas futuras de mejora para implementarlas en el sistema propuesto.

En primer lugar, para clasificar las detecciones, se ha empleado únicamente la información contextual extraída de los segmentadores semánticos. Por lo que se propone introducir, además de la información semántica, la puntuación o *score* de las detecciones para, de esta forma, aumentar el rendimiento del clasificador.

En segundo lugar, en el futuro se podría plantear fijar umbrales distintos en función de la zona en la que aparezca la detección para poder así eliminar falsas detecciones y mejorar la efectividad de reconocimiento de imágenes.

También se podría implementar otro tipo de red neuronal para clasificar las detecciones y de esta forma ver si se obtienen mejores resultados que la red neuronal utilizada en este trabajo.

Bibliografía

[1] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 4, 6, 7, 11

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 4, 5, 6, 7

[3] Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision*, 2011. 4

[4] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 5

[5] Shuchang Zhou, Jia-Nan Wu, Yuxin Wu, and Xinyu Zhou. Exploiting local structures with the kronecker layer in convolutional networks, 2016. 5

[6] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70:41–65, 2018. 6

[7] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 7

[8] Wei Liu, Andrew Rabinovich, and Alexander C. Berg. Parsenet: Looking wider to see better, 2015. 7

[9] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum*, 29(2), 2010. 7

[10] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017. 7, 8

[11] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 8

[12] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 8

[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. 9, 11

[14] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 9

[15] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 9, 11

[16] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE Access*, 2019. 10

[17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 11

[18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018. 11

[19] Grupo de Inteligencia Artificial de Facebook. Documentación de pytorch. 2017. 12

[20] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 14, 15

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr DollÃąr, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. pages 740–755, Cham, 2014. Springer International Publishing. 15

[22] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark. https://github.com/open-mmlab/mmdetection, 2019. 16

[23] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 16